



FACULTY OF TECHNOLOGY AND ENGINEERING

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY

AND RESEARCH

DEPARTMENT OF COMPUTER ENGINEERING

A.Y. 2023-24 [EVEN]

LAB MANUAL

CE259: PROGRAMMING IN PYTHON



Academic Year:2023-2024

Subject Name:PROGRAMMING IN PYTHON

Student Name:PROBIN BHAGCHANDANI

[illegible]

PRACTICAL-1

AIM: A) Introduction to Python Programming. Installation & Configuration of Python. Along with its all-major editors, IDLE, Pycharm, Anaconda, Jupyter, Interpreter etc.

B) Write a python program to calculate simple interest.

IDLE

IDLE (short for Integrated Development and Learning Environment) is an integrated development environment for Python, which has been bundled with the default implementation of the language since 1.5.2b1. It is packaged as an optional part of the Python packaging with many Linux distributions. It is completely written in Python and its GUI toolkit. IDLE is intended to be a simple IDE and suitable for beginners, especially in an educational environment. To that end, it is cross-platform, and avoids feature clutter. According to the included README, its main features are:

- Multi-window text editor with syntax highlighting, autocompletion, smart indent and other.
- Python shell with syntax highlighting.
- Integrated debugger with stepping, persistent breakpoints, and call stack visibility

PyCharm

PyCharm is an integrated development environment (IDE) used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supports web development with Django. PyCharm is developed by the Czech company JetBrains.

It is cross-platform, working on Microsoft Windows, macOS and Linux. PyCharm has a Professional Edition, released under a proprietary license and a Community Edition released under the Apache License. PyCharm Community Edition is less extensive than the Professional Edition.

Anaconda

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS.

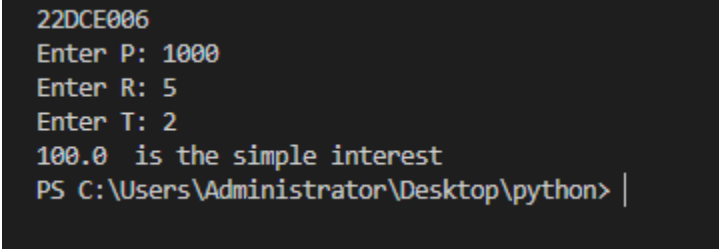


B)

PROGRAM CODE:

```
print("22DCE006")
p=input("Enter P: ")
r=input("Enter R: ")
t=input("Enter T: ")
print(int(int(p)*int(r)*int(t))/100, " is the simple interest")
```

OUTPUT:



```
22DCE006
Enter P: 1000
Enter R: 5
Enter T: 2
100.0 is the simple interest
PS C:\Users\Administrator\Desktop\python> |
```

CONCLUSION: From this practical, I learned about the python fundamentals, installation and configuration of its environment.

Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-2

- AIM:** A) Create a list and apply methods (append, extend, remove, reverse), arrange created list in ascending and descending order.
B) List1 = [1, 2, 3, 4, ["python", "java", "c++", [10,20,30]], 5, 6, 7, ["apple", "banana", "orange"]] . From above list get word “orange” and “Python” & repeat this list five times without using loops.
C) Create a list and copy it using slice function
D) Create a tuple and apply different type of mathematical operation on it (Sum, Maximum, minimum etc.).

A)

PROGRAM CODE:

```
print("22DCE006")
list=[10,20,40,80]
print(list)
print("Append Function")
list.append(77)
print(list)
list2=[11,22,33]
print("Extend Function")
list.extend(list2)
print(list)
print("Remove Function")
list.remove(77)
print(list)
print("Reverse Function")
list.reverse()
print(list)
print("Ascending Sorting")
list.sort()
print(list)
print("Descending Sorting")
list.sort(reverse=True)
print(list)
```

OUTPUT:


```

PS D:\python> python trial.py
22DCE006
[10, 20, 40, 80]
Append Function
[10, 20, 40, 80, 77]
Extend Function
[10, 20, 40, 80, 77, 11, 22, 33]
Remove Function
[10, 20, 40, 80, 11, 22, 33]
Reverse Function
[33, 22, 11, 80, 40, 20, 10]
Ascending Sorting
[10, 11, 20, 22, 33, 40, 80]
Descending Sorting
[80, 40, 33, 22, 20, 11, 10]
PS D:\python> |

```

B)

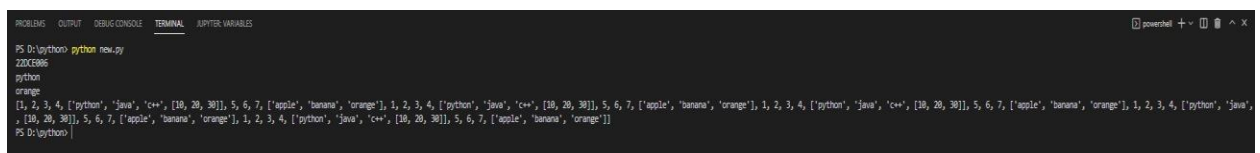
PROGRAM CODE:

```

print("22DCE006")
List1 = [1, 2, 3, 4, ["python", "java", "c++", [10,20,30] ], 5, 6, 7, [ "apple" , "banana" , "orange" ]
]
print(List1[4][0])
print(List1[8][2])
repeatedlist=List1*5
print(repeatedlist)

```

OUTPUT:



```

PS D:\python> python new.py
22DCE006
['python', 'java', 'c++', [10, 20, 30], 5, 6, 7, ['apple', 'banana', 'orange'], 1, 2, 3, 4, ['python', 'java', 'c++', [10, 20, 30]], 5, 6, 7, ['apple', 'banana', 'orange'], 1, 2, 3, 4, ['python', 'java', 'c++', [10, 20, 30]], 5, 6, 7, ['apple', 'banana', 'orange'], 1, 2, 3, 4, ['python', 'java', 'c++', [10, 20, 30]], 5, 6, 7, ['apple', 'banana', 'orange'], 1, 2, 3, 4, ['python', 'java', 'c++', [10, 20, 30]], 5, 6, 7, ['apple', 'banana', 'orange']]
PS D:\python>

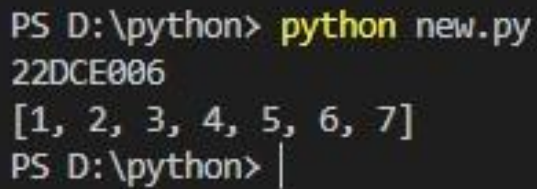
```

C)

PROGRAM CODE:

```
print("22DCE006")
list=[1,2,3,4,5,6,7]
x=slice(0,7)
list2=list[x]
print(list2)
```

OUTPUT:



```
PS D:\python> python new.py
22DCE006
[1, 2, 3, 4, 5, 6, 7]
PS D:\python> |
```

D)

PROGRAM CODE:

```
print("22DCE006")
a=(1,2,3,4,5,6,7)
print(a)
print("Sum Operation:")
print(sum(a))
print("Maximum Operation:")
print(max(a))
print("Minimum Operation:")
print(min(a))
```

OUTPUT:

```
PS D:\python> python new.py
22DCE006
(1, 2, 3, 4, 5, 6, 7)
Sum Operation:
28
Maximum Operation:
7
Minimum Operation:
1
PS D:\python> |
```

CONCLUSION: In this practical I learned various method like append, extend, remove, reverse, sort in python and how to use slice method and maximum and minimum method in python.

Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-3

AIM:

A) String Operations:

- Reverse a string, replace string with other string, merge two strings
- Find character is in string or not without using loops
- Split string into multiple words & characters

B) Dictionaries Operations:

- Apply “Update, Delete, clear, popitem, pop, get, keys and values” operation in dictionary.
- Create 3 dictionaries and merge them into 1 dictionary

PROGRAM CODE:

```
import re
print("22DCE006\n")
str1="hello" [::-1]
print("Reverse String="+str1)
str2="Hello World"
str3 = str2.replace("Hello", "Bye")
print("Replacing the String = "+str3)
str4="Charusat"
str5="University"
str6 = str4 + ' ' + str5
print("Merging two Strings = "+str6)
str7="Python Lab"
print("The 'o' character is present in the string:", 'o' in str7)
str8="Charsuat University"
result = re.findall(r"[\w]+", str8)
print("After splitting the string: ", result)

Dict= { "movie" : "Hero", "theme" : "fiction", "year" : "2021" }
print(Dict)
Dict.update({"bg":"dark"})
print("\nUpdating the dictionary: "+str(Dict))
del Dict["bg"]
print("After deleting: "+str(Dict))
print("Using keys() method: "+str(Dict.keys()))
print("Using values() method: "+str(Dict.values()))
Dict.popitem()
```

```

print("Using popitem: "+str(Dict))
print("Using get() method: "+str(Dict.get("movie")))
Dict.pop("theme")
print("Dictionary after using pop: "+str(Dict))
Dict2 = { "cast" : "Viraj" }
Dict3 = { "review" : "7-star" }
Dict.update(Dict2)
Dict.update(Dict3)
print("Merge Operation "+str(Dict))

```

OUTPUT:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

PS C:\Users\Administrator\Desktop\python> python prac.py
22DCE006

Reverse String=olleh
Replacing the String = Bye World
Merging two Strings = Charusat University
The 'o' character is present in the string: True
After splitting the string: ['Charsuat', 'University']
{'movie': 'Hero', 'theme': 'fiction', 'year': '2021'}

Updating the dictionary: {'movie': 'Hero', 'theme': 'fiction', 'year': '2021', 'bg': 'dark'}
After deleting: {'movie': 'Hero', 'theme': 'fiction', 'year': '2021'}
Using keys() method: dict_keys(['movie', 'theme', 'year'])
Using values() method: dict_values(['Hero', 'fiction', '2021'])
Using popitem: {'movie': 'Hero', 'theme': 'fiction'}
Using get() method: Hero
Dictionary after using pop: {'movie': 'Hero'}
Merge Operation {'movie': 'Hero', 'cast': 'Viraj', 'review': '7-star'}
PS C:\Users\Administrator\Desktop\python> 

```

CONCLUSION: In this practical I learned various operations that can be performed on strings and also about dictionary operations.

Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-4

AIM: Write python programs by declaring a function

A) Found which grade student will get based on SGPA.

B) Find max from three numbers

C) Calculate number of Uppercase and lowercase letters of string given by user

D) Find a Square of a given list using lambda function

E) Enter value from user and print multiplication table

F) Create a list by user given value and make sum of it using loop

G) Use comprehension method

- **Create a two separate list of even and odd numbers from 1 to 50**
- **Get value which are divided by 5 from 1 to 100**

A)

Program Code:

```
print("22DCE006\n")
maths=float(input("Enter the marks of maths:"));
science=float(input('enter the marks of science:'));
English=float(input('enter the marks of English:'));
Hindi=float(input('Enter the marks of Hindi:'));
grade='\0'
total=maths+science+English+Hindi
avg=total/4
if(avg>=90):
    grade ='A'
elif((avg>=80) and (avg<90)):
    grade='B'
elif((avg>=70) and (avg<80)):
    grade='C'
elif((avg>=60) and (avg<70)):
    grade='D'
elif((avg>=50) and (avg<60)):
    grade='E'
else:
    grade='F'
print("your grade is",grade)
```

Output:

```
PS D:\Probin's Work\Python> python new.py
22DCE006

Enter the marks of maths:97
enter the marks of science:99
enter the marks of English:98
Enter the marks of Hindi:94
your grade is A
PS D:\Probin's Work\Python> |
```

B)

Program Code:

```
print("22DCE006\n")
a = int(input('Enter first number : '))
b = int(input('Enter second number : '))
c = int(input('Enter third number : '))
max = 0
if a > b and a > c :
    max = a
elif b > c :
    max = b
else :
    max = c
print(max, "is the maximum of three numbers.")
```

Output:

```
PS D:\Probin's Work\Python> python new.py
22DCE006

Enter first number : 77
Enter second number : 54
Enter third number : 99
99 is the maximum of three numbers.
PS D:\Probin's Work\Python> |
```

C)

Program Code:

```
print("22DCE006\n")
def count_letters(string):
    uppercase_count = 0
    lowercase_count = 0

    for char in string:
```

```
if char.isupper():
    uppercase_count += 1
elif char.islower():
    lowercase_count += 1

return uppercase_count, lowercase_count
```

```
string = input("Enter a string: ")
uppercase, lowercase = count_letters(string)
print(f"Uppercase letters: {uppercase}")
print(f"Lowercase letters: {lowercase}")
```

Output:

```
PS D:\Probin's Work\Python> python new.py
22DCE006

Enter a string: Hello
Uppercase letters: 2
Lowercase letters: 3
PS D:\Probin's Work\Python> |
```

D)

Program Code:

```
print("22DCE006\n")
number = [7,8,9,10]
print("Square every number are:")
square_number = list(map(lambda x: x ** 2, number))
print(square_number)
```

Output:

```
PS D:\Probin's Work\Python> python new.py
22DCE006

Square every number are:
[49, 64, 81, 100]
PS D:\Probin's Work\Python> |
```

E)

Program Code:

```
print("22DCE006\n")
n = int(input("Enter a number: "))
for i in range(1,11):
    print(n,'x',i,'=',n*i)
```


Output:

```
PS D:\Probin's Work\Python> python new.py
22DCE006

Enter a number: 7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
PS D:\Probin's Work\Python> |
```

F)

Program Code:

```
print("22DCE006\n")
LIST=[]
n=int(input("ENter the number of element you want to insert in the list: "))
for i in range(0,n):
    e=int(input())
    LIST.append(e)
print("LIST IS :",LIST)
sum=0
for i in range(0,len(LIST)):
    sum=sum+LIST[i]
print("Sum of the list is",sum)
```

Output:

```
PS D:\Probin's Work\Python> python new.py
22DCE006

Enter the number of element you want to insert in the list: 5
1
2
3
4
5
LIST IS : [1, 2, 3, 4, 5]
Sum of the list is 15
PS D:\Probin's Work\Python> |
```

G)

Program Code:

```
print("22DCE006\n")
LIST1=[]
LIST2=[]
for i in range(1,51):
    if i&1:
        LIST1.append(i)
    else:
        LIST2.append(i)
print("The list of the even number is",LIST2)
print("The list of the odd number is",LIST1)
list3=[]
for i in range(1,101):
    if i%5==0:
        list3.append(i)
print("The number divisible by the 5 in range 1 to 100 are",list3)
```

Output:

```
PS D:\Probin's Work\Python> python new.py
22DCE006

The list of the even number is [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50]
The list of the odd number is [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49]
The number divisible by the 5 in range 1 to 100 are [5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]
PS D:\Probin's Work\Python> |
```

Conclusion: In this practical we learnt about the different type of the fuction like the lambda fuction, append and how to create the list in which user can enter the data,if else statement in the python and its usage for manipulating the list .

Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-5

AIM:

- A) Create a class employee and display employee details
- B) From above create class and count number of employee and display a salary amount, if the salary is raised to 1.04%.
- C) Fetch children class details using different types of inheritance (Single, Multilevel, and Multiple) With constructor
- D) Find who will be first among two students using polymorphism.

A)

Program Code:

```
print("22DCE006\n")
class emp:
    c=0
    def __init__(self,name,e_id,salary):
        self.name=name
        self.e_id=e_id
        self.salary=salary
        emp.c+=1
    def disp(self):
        print("The name of the employee is : ",self.name)
        print("The Employee id is : ",self.e_id)
        print("The salary is : ",self.salary)
p1=emp("raj",45,45000)
p1.disp()
p2=emp("ram",50,75000)
p2.disp()
```

Output:

```
PS D:\Probin's Work\Python> python new.py
22DCE006

The name of the employee is : raj
The Employee id is : 45
The salary is : 45000
The name of the employee is : ram
The Employee id is : 50
The salary is : 75000
PS D:\Probin's Work\Python> |
```

B)

Program Code:

```
print("22DCE006\n")
class emp:
    c=0
    def __init__(self,name,e_id,salary):
        self.name=name
        self.e_id=e_id
        self.salary=salary
        emp.c+=1
    def disp(self):
        print("The name of the employee is : ",self.name)
        print("The Employee id is : ",self.e_id)
        print("The salary is : ",self.salary)
    def inc(self):
        inc=(self.salary*0.0104)+self.salary
        print("Acutal salary is : ",self.salary)
        print("The incremented salary is : ",inc)

p1=emp("raj",45,45000)
p1.disp()
p1.inc()
```

Output:

```
PS D:\Probin's Work\Python> python new.py
22DCE006

The name of the employee is : raj
The Employee id is : 45
The salary is : 45000
Acutal salary is : 45000
The incremented salary is : 45468.0
PS D:\Probin's Work\Python> |
```

C)

Program Code:

```
print("22DCE006\n")
# Single Inheritance
class Parent:
    def __init__(self, name):
        self.name = name

    def display(self):
        print("Parent Class")

class Child(Parent):
    def __init__(self, name, age):
        super().__init__(name)
        self.age = age

    def display_child(self):
        print("Child Class")
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")

# Multilevel Inheritance
class GrandParent:
    def __init__(self, name):
        self.name = name

    def display(self):
        print("GrandParent Class")

class Parent(GrandParent):
    def __init__(self, name, age):
        super().__init__(name)
        self.age = age

    def display_parent(self):
        print("Parent Class")
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")

class Child(Parent):
    def __init__(self, name, age, grade):
        super().__init__(name, age)
        self.grade = grade

    def display_child(self):
        print("Child Class")
```

```
print(f"Name: {self.name}")
print(f"Age: {self.age}")
print(f"Grade: {self.grade}")
```

Multiple Inheritance

```
class Father:
    def __init__(self, name):
        self.name = name

    def display_father(self):
        print("Father Class")
        print(f"Name: {self.name}")
```

```
class Mother:
    def __init__(self, age):
        self.age = age

    def display_mother(self):
        print("Mother Class")
        print(f"Age: {self.age}")
```

```
class Child(Father, Mother):
    def __init__(self, name, age, grade):
        super().__init__(name)
        Mother.__init__(self, age)
        self.grade = grade

    def display_child(self):
        print("Child Class")
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")
        print(f"Grade: {self.grade}")
```

```
# Single Inheritance Example
child1 = Child("Rajesh", 10, 7)
child1.display_child()
```

```
# Multilevel Inheritance Example
child2 = Child("Suresh", 12, 6)
child2.display_child()
```

```
# Multiple Inheritance Example
child3 = Child("Harish", 8, 3)
child3.display_father()
child3.display_mother()
child3.display_child()
```

Output:

```
PS D:\Probin's Work\Python> python new.py
22DCE006

Child Class
Name: Rajesh
Age: 10
Grade: 7
Child Class
Name: Suresh
Age: 12
Grade: 6
Father Class
Name: Harish
Mother Class
Age: 8
Child Class
Name: Harish
Age: 8
Grade: 3
PS D:\Probin's Work\Python> |
```

D)

Program Code:

```
print("22DCE006\n")
class Student:
    def __init__(self, name, id):
        self.name = name
        self.id = id

    def __lt__(self, other):
        return self.id < other.id

student1 = Student("Rajesh", 1)
student2 = Student("Neeraj", 2)

if student1 < student2:
    print(f"{student1.name} will be first")
else:
    print(f"{student2.name} will be first")
```

Output:

```
PS D:\Probin's Work\Python> python new.py
22DCE006

Rajesh will be first
PS D:\Probin's Work\Python> |
```

Conclusion: From this practical we learned the concept of class and constructors. Also learned about the different types of inheritance , its usage and polymorphism.

Signature:

Grade:

Remarks by the Staff:

PRACTICAL-6

AIM:

Consider an example of declaring the examination result. Design three classes: Student, Exam, and Result. The Student class has data members such as those representing rollNumber, Name, etc. Create the class Exam by inheriting Student class. The Exam class adds fields representing the marks scored in six subjects. Derive Result from the Exam class, and it has its own fields such as total marks. Write an interactive program to model this relationship.

Program Code:

```
print("22DCE006\n")
class Student:
    def __init__(self, roll_number, name):
        self.roll_number = roll_number
        self.name = name

    def display(self):
        print(f"Roll Number: {self.roll_number}")
        print(f"Name: {self.name}")

class Exam(Student):
    def __init__(self, roll_number, name, marks):
        super().__init__(roll_number, name)
        self.marks = marks

    def display_marks(self):
        print("Marks:")
        for subject, marks in self.marks.items():
            print(f"{subject}: {marks}")

class Result(Exam):
    def __init__(self, roll_number, name, marks, total_marks):
        super().__init__(roll_number, name, marks)
        self.total_marks = total_marks

    def display_result(self):
        self.display()
        self.display_marks()
        print(f"Total Marks: {self.total_marks}")
        print("Percentage obtained by the student is : " + str(self.total_marks/6))

# Interactive program
```

```
roll_number = input("Enter Roll Number: ")
name = input("Enter Name: ")

marks = {}
subjects = ["Subject1", "Subject2", "Subject3", "Subject4", "Subject5", "Subject6"]
for subject in subjects:
    marks[subject] = float(input(f"Enter marks for {subject}: "))

total_marks = sum(marks.values())

result = Result(roll_number, name, marks, total_marks)
result.display_result()
```

Output:

```
PS D:\Probin's Work\Python> python new.py
22DCE006

Enter Roll Number: 31
Enter Name: david
Enter marks for Subject1: 95
Enter marks for Subject2: 97
Enter marks for Subject3: 98
Enter marks for Subject4: 99
Enter marks for Subject5: 93
Enter marks for Subject6: 96
Roll Number: 31
Name: david
Marks:
Subject1: 95.0
Subject2: 97.0
Subject3: 98.0
Subject4: 99.0
Subject5: 93.0
Subject6: 96.0
Total Marks: 578.0
Percentage obtained by the student is : 96.33333333333333
PS D:\Probin's Work\Python> |
```

Conclusion: From this practical we learned the key concepts such inheritance, polymorphism, encapsulation, and the use of interactive programming techniques. With these concepts we can help in designing and implementing effective code structures.

Signature:

Grade:

Remarks by the Staff: